

Shortest paths between shortest paths and independent sets^{*}

Marcin Kamiński^{1**}, Paul Medvedev², and Martin Milanič^{3***}

¹Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium
Marcin.Kaminski@ulb.ac.be

²Department of Computer Science, University of Toronto, Toronto, Canada
pashadag@cs.toronto.edu

³FAMNIT and PINT, University of Primorska, Koper, Slovenia
martin.milanic@upr.si

Abstract. We study problems of reconfiguration of shortest paths in graphs. We prove that the shortest reconfiguration sequence can be exponential in the size of the graph and that it is NP-hard to compute the shortest reconfiguration sequence even when we know that the sequence has polynomial length. Moreover, we also study reconfiguration of independent sets in three different models and analyze relationships between these models, observing that shortest path reconfiguration is a special case of independent set reconfiguration in perfect graphs, under any of the three models. Finally, we give polynomial results for restricted classes of graphs (even-hole-free and P_4 -free graphs).

1 Introduction

One of the biggest impacts of algorithmic graph theory has been its usefulness in modeling real-world problems, where the domain of the problem is modeled as a graph and the constraints on the solution define feasible solutions. For example, consider the problem of routing a certain commodity between two nodes in a transportation network, using as few hops as possible. The transportation network can be modeled as a graph, each route can be modeled as a path, and the feasible solutions are all the shortest paths between the two nodes. Traditionally, the real-world user first defines a problem instance and then uses an algorithm to find a feasible solution which she then “implements” in the real world. However, some real-world situations do not follow this simple paradigm and are more dynamic because they allow the solution to “evolve” over time. For example, consider the situation where the commodity is already being transferred along a shortest route, but the operator has been instructed to use a different route, which is also a shortest path. She can physically switch the

^{*} A preliminary version of this work has been accepted to IWOCA 2010, 21st International Workshop on Combinatorial Algorithms, London, 26-28 July 2010 [14].

^{**} Chargé de Recherches du FRS-FNRS.

^{***} Supported in part by “Agencija za raziskovalno dejavnost Republike Slovenije”, research program P1-0285.

route only one node at a time, but does not wish to interrupt the transfer. Thus, she would like to switch between the two routes in as few steps as possible, while maintaining a shortest path route at every intermediate step.

In general, this type of situation gives rise to a *reconfiguration* framework, where we consider an algorithmic problem \mathcal{P} and a way of transforming one feasible solution of an instance I of \mathcal{P} to another (*reconfiguration rule*). Given two feasible solutions s_1, s_k of I , we want to find a *reconfiguration sequence* s_1, \dots, s_k such that each s_i ($1 \leq i \leq k$) is a feasible solution of I , and the transition between s_i and s_{i+1} is allowed by the reconfiguration rule. An alternate definition is via the *reconfiguration graph*, where the vertices are the feasible solutions of I , and two solutions are adjacent if and only if one can be obtained from the other by the reconfiguration rule. The reconfiguration sequence is then a path between s_1 and s_k in the reconfiguration graph. We can then ask for the shortest reconfiguration sequence, or, in the *reconfigurability problem*, to simply check if the two solutions are *reconfigurable* (i.e., if such a sequence exists).

The reconfiguration framework has recently been applied in a number of settings, including vertex coloring [4, 5, 3, 2], list-edge coloring [13], clique, set cover, integer programming, matching, spanning tree, matroid bases [12], block puzzles [11], independent set [11, 12], and satisfiability [10]. In the well-studied vertex coloring problem, for example, we are given two k -colorings of a graph, and the reconfiguration rule allows to change the color of a single vertex. In a different example, we are given two independent sets, which we imagine to be two sets of tokens placed on the vertices, and the reconfiguration rule is to slide a single token along an edge (*token sliding*).

Though the complexities of each of the many reconfiguration problems may each be studied independently, a fundamental question is whether there exists any systematic relationship between the complexity of the original problem and that of its reconfigurability problem. To this end, current studies have revealed a pattern where most “natural” problems in P have their reconfigurability problems in P as well, while problems whose reconfigurability versions are at least NP-hard are NP-complete. For example, spanning tree, matching, and matroid problems in general (all in P) lead to polynomially solvable reconfigurability problems, while the reconfigurability of independent set, set cover, and integer programming (all NP-complete) are PSPACE-complete [12]. Another example is satisfiability, where Gopalan et al. [10] showed that reconfigurability instances arising from tight relations—a class for which it is easy to determine if the formula is satisfiable—can be solved in linear time; on the other hand, reconfigurability is PSPACE-complete for the class of formulas arising from non-tight relations.

Ito et al. [12] have conjectured that this relationship is not true in general, and that there exist problems in P which give rise, in a natural way, to NP-hard reconfigurability problems. Indeed, the problem of deciding whether two k -colorings are reconfigurable is PSPACE-complete for (i) bipartite graphs and $k \geq 4$, and (ii) planar graphs, for $4 \leq k \leq 6$ [2]. Clearly, 4-coloring of bipartite or planar graphs is in P. However, these are not “natural” problems in the sense that the colorings are not optimal. It is interesting to ask if there exists a “natural” problem in P whose reconfiguration version is NP-hard.

Another systematic relationship that has been pursued is between the complexity of a reconfigurability problem and the diameter of the reconfiguration graph. When the diameter is polynomial, a reconfiguration sequence is a trivial certificate for the reconfigurability of two instances, guaranteeing that the problem is in NP. However, current evidence further suggests that for reconfigurability problems that are solvable in polynomial time, the diameter is also polynomial. In the study of k -coloring, it was found that for $k \leq 3$, the reconfigurability problem is solvable in polynomial time and the diameter of the reconfiguration graph is at most quadratic in the number of vertices of the colored graph. For satisfiability, the formulas built from tight relations (whose reconfigurability is polynomial) lead to reconfiguration graphs with linear diameter [10]. We are not aware of any natural problems with the property that the diameter can be exponential while reconfigurability can be decided in polynomial time¹; however, such an example, if found, would indicate that the diameter cannot serve as a reliable indicator of the reconfigurability complexity.

In this paper, we introduce the reconfiguration version of the shortest path problem (Section 2), which arises naturally, such as in the routing example above. We show that the reconfiguration graph can have exponential diameter, implying that the shortest path reconfiguration problem probably breaks one of the two established patterns described above. On the one hand, if reconfigurability of shortest paths can be decided in polynomial time, then it is the first example of a reconfigurability problem in P with exponential diameter. On the other hand, if it is NP-hard, it is the first example of a “natural” problem in P whose reconfigurability version is NP-hard. For these reasons, we believe that shortest path reconfiguration is an important problem to study, not only for its practical application but also for our understanding of the systematic relationship between the hardness of a problem, the diameter of its reconfiguration graph, and the hardness of its reconfigurability problem. Towards this end, we give a non-trivial reduction from SAT to show that it is NP-hard to find the shortest reconfiguration sequence between two shortest paths

¹ For a very artificial one, consider the problem in which instances are n -bit words and two instances are adjacent if they differ by 1 modulo 2^n . The diameter of the reconfiguration graph is 2^{n-1} but all pairs of instances are reconfigurable.

(however, the complexity status of the reconfigurability problem remains open).

We also study reconfiguration of independent sets, where, unlike many other problems, there is more than one natural reconfiguration rule. In particular applications, for example, a threshold is specified that bounds the cardinality of the intermediate feasible solutions. Based on this idea, Ito et al. [12] considered an alternative to token sliding called *token addition and removal*, where one is allowed to either add or remove a token as long as there are at least $k - 1$ tokens at any given time, for some k . In this paper, we introduce *token jumping*, where one is allowed to move a single token to any other vertex. The token jumping reconfiguration graph is often easier to analyze than the token addition and removal one, since the cardinalities of two adjacent token sets are always the same. However, we show that the two models are polynomially equivalent, allowing for an easier way to analyze token addition and removal reconfiguration graphs (Section 3).

Finally, we show that reconfiguration of independent sets is a generalization of the reconfiguration of shortest paths; our hardness result for shortest paths then implies that it is NP-hard to find the shortest reconfiguration sequence between two independent sets, even in perfect graphs (Section 4). We also identify two restricted graph classes where reconfigurability is easy – even-hole-free graphs under token jumping, for which the reconfiguration graph is always connected, and P_4 -free graphs under token sliding (Section 5).

2 Shortest path reconfiguration

We define the reconfiguration rule for shortest paths in the natural way: two shortest (s, t) -paths are adjacent in the reconfiguration graph of shortest (s, t) -paths if and only if they differ, as sequences, in exactly one vertex.

2.1 Instances with exponential diameter

We now present a family of graphs G^k whose size is linear in k but the diameter of the reconfiguration graph is $\Omega(2^k)$. The graph G^1 contains vertices $\{x_i^1 \mid 1 \leq i \leq 7\} \cup \{y_i^1 \mid 1 \leq i \leq 6\} \cup \{s, t\}$ and edges $\{(x_i^1, y_i^1), (x_{i+1}^1, y_i^1), (y_i^1, t) \mid i \leq 6\} \cup \{(s, x_i^1) \mid 1 \leq i \leq 7\}$. The graph G^k is defined recursively with vertices $\{x_i^k \mid 1 \leq i \leq 7\} \cup \{y_i^k \mid 1 \leq i \leq 6\} \cup V(G^{k-1})$ and the edges $\{(x_i^k, y_i^k), (x_{i+1}^k, y_i^k) \mid i \leq 6\} \cup \{(y_i^k, x_j^{k-1}) \mid i \in \{1, 3, 5\}, j \leq 7\} \cup \{(y_2^k, x_1^{k-1}), (y_4^k, x_7^{k-1}), (y_6^k, x_1^{k-1})\} \cup E(G^{k-1} \setminus \{s\}) \cup \{(s, x_i^k) \mid 1 \leq i \leq 7\}$ (see Figure 1). Let $p_b^k = s, x_1^k, y_1^k, \dots, x_1^1, y_1^1, t$, and let $p_e^k = s, x_7^k, y_6^k, x_1^{k-1}, x_1^{k-1}, \dots, x_1^1, y_1^1, t$. We will consider the problem of reconfiguring p_b^k to p_e^k in G^k .

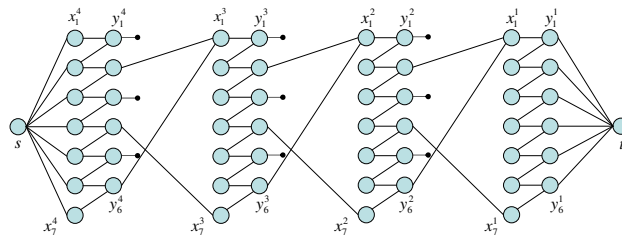


Fig. 1. The graph G^k for $k = 4$, where the reconfiguration distance between $p_b^k = s, x_1^k, y_1^k, \dots, x_1^1, y_1^1, t$ and $p_e^k = s, x_7^k, y_6^k, x_1^{k-1}, x_1^{k-1}, \dots, x_1^1, y_1^1, t$ is $\Theta(2^k)$. An edge with a circle end means that the vertex is connected to all the vertices in the next layer.

Lemma 1. *Let p be a shortest path in G^k that goes through y_1^k , and let q be a path that goes through y_6^k . Then the reconfiguration distance between p and q is at least $9(2^k - 1)$.*

Proof. We prove by induction on k , where the base case is clear. Let $\rho = p_1, \dots, p_n$ be the shortest reconfiguration sequence between p and q . First, let i' be the smallest integer such that $p_{i'+1}$ contains y_4^k , and let $i \leq i'$ be the smallest integer such that every path $p_i, \dots, p_{i'}$ contains y_3^k . By construction, we know that p_{i-1} , and hence p_i , contains y_1^{k-1} and $p_{i'+1}$, and hence $p_{i'}$, contains y_6^{k-1} . Hence, by the induction hypothesis, the length of this first phase, $i' - i + 1$, is at least $9(2^{k-1} - 1)$.

Next, let j' be the smallest integer such that $p_{j'+1}$ contains y_6^k , and let $j \leq j'$ be the smallest integer such that every path $p_j, \dots, p_{j'}$ contains y_5^k . By construction, we know that p_{j-1} , and hence p_j , contains y_6^{k-1} and $p_{j'+1}$, and hence $p_{j'}$, contains y_1^{k-1} . Hence, by the induction hypothesis, the length of this second phase, $j' - j + 1$, is at least $9(2^{k-1} - 1)$.

Observe from the graph construction that ρ must always visit y_{x-1}^k before visiting y_x^k , hence $i' < j$, and so the length of ρ is at least the sum of the two phases plus the moves of the first and second vertex necessary to percolate y_1^k down to y_6^k , proving the lemma. \square

On the other hand, there exists an asymptotically matching lower bound:

Lemma 2. *The reconfiguration distance between p_b^k and p_e^k is at most $11(2^k - 1)$.*

Proof. We prove by induction on k , where the base case is clear. It will be helpful to formally treat a reconfiguration sequence not as a sequence of paths but as a sequence of vertices, each one representing the switched vertex at that step. Applying the induction hypothesis, let ρ be the shortest reconfiguration sequence in G^{k-1} , and let $rev(\rho)$ be that sequence

in the reverse direction (from p_e^{k-1} to p_b^{k-1}). We construct the sequence as $\rho' = x_2^k, y_2^k, x_3^k, y_3^k, \rho, x_4^k, y_4^k, x_5^k, y_5^k, rev(\rho), x_6^k, y_6^k, x_7^k$. This sequence of moves reconfigures p_b^k into p_e^k with the number of steps satisfying the lemma. \square

We therefore have the following theorem:

Theorem 1. *The reconfiguration distance in G^k between p_b^k and p_e^k is $\Theta(2^k)$.*

2.2 NP-Hardness of MIN-SPR

Given (G, s, t, p_b, p_e, k) , where p_b and p_e are shortest (s, t) -paths and k is an integer, the MIN-SPR problem is to determine whether there is a reconfiguration sequence between p_b and p_e of length at most k . Let ϕ be a formula with variables x_1, \dots, x_n and clauses C_1, \dots, C_m . We will create an instance $(G_\phi, s, t, p_b, p_e, 2m(n+2))$ and show that ϕ is satisfiable if and only this instance is in MIN-SPR. For ease of presentation, the graph G_ϕ will be directed. However, our result holds for undirected graphs because the directed shortest (s, t) -paths in G_ϕ are exactly the shortest paths in the undirected version of G_ϕ .

For every variable x_i and its possible value $vs \in \{0, 1\}$, we build a gadget $G(i, vs)$. The vertex set is $\{v(i, vs, cs, j) \mid cs \in \{0, 1\}, 1 \leq j \leq 2m\}$. The values i , vs , cs , and j for a vertex are referred to as its *level*, *v-state*, *c-state*, and *depth*, and denoted by $l(v)$, $vs(v)$, $cs(v)$, and $d(v)$, respectively. For every $1 \leq j \leq 2m - 1$, and every cs , there is an edge from $v(i, vs, cs, j)$ to $v(i, vs, cs, j + 1)$. For all $1 \leq j \leq m - 1$, there is an edge from $v(i, vs, 0, 2j)$ to $v(i, vs, 1, 2j + 1)$, and from $v(i, vs, 1, 2j)$ to $v(i, vs, 0, 2j + 1)$. We also add edges, called formula edges, that are formula dependent. For all j , if $x_i = vs$ satisfies C_j , we add an edge from $v(i, vs, 1, 2j - 1)$ to $v(i, vs, 0, 2j)$. This gadget is shown in Figure 2A.

We now connect some of these gadgets together. The gadgets we connect are $G(i, vs)$ to $G(i + 1, 0)$ and to $G(i + 1, 1)$, for all $i \leq n - 1$ and all vs . Given two gadgets, $G(i, vs)$ and $G(i', vs')$, the meaning of connecting $G(i, vs)$ to $G(i', vs')$ is given as follows (shown in Figure 2BC). For all $j \leq 2m - 1$ and cs , there is an edge from $v(i', vs', cs, j - 1)$ to $v(i, vs, cs, j)$. Also, for all $j \leq m - 1$ and cs , there is an edge from $v(i', vs', cs, 2j)$ to $v(i, vs, 1 - cs, 2j + 1)$.

We next add a begin and end gadget to the graph, consisting of vertices beg_j and end_j , respectively, for $1 \leq j \leq 2m$. These are connected in a path, with edges (beg_j, beg_{j+1}) and (end_j, end_{j+1}) for $j \leq 2m - 1$. The level of the vertices in the begin (end) gadget is 0 ($i + 1$), the c-state is 0 (1), and the depth of beg_j or end_j is j . For all vs , $j \leq 2m - 1$, there is an edge from $v(1, vs, 0, j)$ to beg_{j+1} , and from end_j to $v(n, vs, 1, j + 1)$.

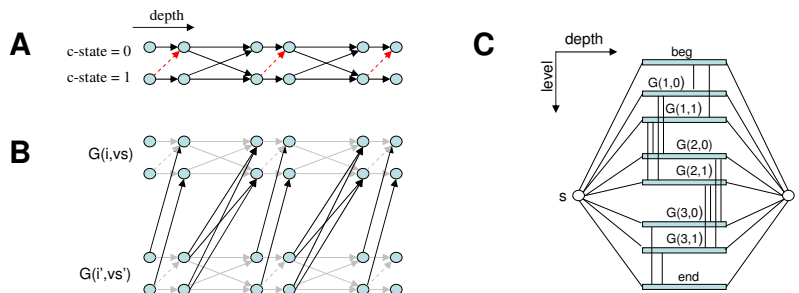


Fig. 2. The reduction from a formula ϕ to a graph G_ϕ for the case of three clauses and three variables. Panel A shows the internal connections of a gadget, with the potential formula edges that depend on ϕ given in red (dashed). Panel B shows the way we connect two given gadgets, while C shows the structure of the whole graph. Each of the rectangles represents a gadget, with the lines showing which parts are connected together.

Finally, we add a s and t vertex to the graph, and make an edge from s to every depth 1 vertex, and from every depth $2m$ vertex to t . The depth of $s(t)$ is defined to be 0 ($2m + 1$). We call the resulting directed graph G_ϕ . Let $p_b = s, beg_1, \dots, beg_{2m}, t$ and $p_e = s, end_1, \dots, end_{2m}, t$ be two paths in this graph. Then, $(G_\phi, s, t, p_b, p_e, 2m(n + 2))$ is the instance of the MIN-SPR problem that we will consider here.

The intuition behind the reduction is that in order for the path to percolate down from p_b to p_e in a minimal number of steps, it must pass consecutively through exactly one of $G(i, 0)$ or $G(i, 1)$ for every variable x_i . The choice of which one corresponds to assigning x_i the corresponding value. Furthermore, each shortest path that goes through a gadget can visit the vertex at depth $2j$ with a c-state of 0 or 1. This corresponds to having the j^{th} clause satisfied or not. Initially, the path goes only through vertices with c-state 0, and the only way to switch the c-state at a given depth is via a formula edge. By going through a gadget $G(i, vs)$, there is an opportunity to use the formula edges to switch the c-state of all clauses that $x_i = vs$ would satisfy. In order to reach the final path p_e , the c-state of all the vertices must be 1, hence all the clauses must be satisfied.

First, we will show that the reduction is sound. Each edge (a, b) is considered to be either *odd* or *even*, depending on the parity of $d(a)$. We call edges that connect vertices on the same level (exactly those that belong to the same gadget) as *intra-level*, while the edges that connect vertices on different levels are called *inter-level*. We say that a reconfiguration sequence *visits* a vertex if there exists $p \in \rho$ that contains that vertex.

Fact 1 *Let $e = (a, b)$ be an edge in G_ϕ . The following follows directly from construction:*

1. $l(b) \leq l(a) \leq l(b) + 1$.

2. If e is an intra-level odd edge, $cs(a) = 0$ implies that $cs(b) = 0$.
3. If e is an inter-level odd edge, then $cs(a) = cs(b)$.
4. If e is a non-formula odd edge, then $cs(a) = cs(b)$.
5. If e is intra-level, then $vs(a) = vs(b)$.

These facts about G_ϕ capture most of the properties of the reduction that are needed to prove completeness and soundness. We will first need some definitions. Let $p = s, v_1, \dots, v_{2m}, t$ be a shortest path and consider an arbitrary move that switches v_i with v'_i . The *move graph* is the subgraph induced by $v_{i-1}, v_i, v'_i, v_{i+1}$, referred to by the tuple $(v_{i-1}, v_i, v'_i, v_{i+1})$. There are two kinds of moves, *odd* and *even*, depending on the parity of $d(v_i)$.

Lemma 3. *The length of a reconfiguration sequence is at least $2m(n+2)$. Moreover, each move in an sequence that has this length must either increase the c-state or the level of the switched vertex, but not both.*

Proof. Let $\Phi(p) = \sum_{v \in p \setminus \{s, t\}} l(v) + cs(v)$ be a potential function counting the sum levels and c-states for vertices in a path. Consider an arbitrary move m and its move graph (a, b, c, d) . We will first show that $\Delta\Phi(m) = l(c) - l(b) + cs(c) - cs(b) \leq 1$. First consider the case that the move is odd. Applying Fact 1.1 to the edges (b, d) and (c, d) , we get that $l(c) \leq l(b) + 1$. Moreover, the only way to have $l(c) = l(b) + 1$ is for (b, d) to be intra-level and (c, d) to be inter-level. In this case, and further supposing $cs(b) = 0$, Fact 1.2 implies that $cs(d) = 0$, and Fact 1.3 implies that $cs(c) = 0$. These facts imply $\Delta\Phi(m) \leq 1$. A similar argument, applied to the edges (a, b) and (a, c) , holds for the case m is even. Combined with the fact that $\Phi(p_b) = 0$ and $\Phi(p_e) = 2m(n+2)$, the lemma follows. \square

Lemma 4. *No path can contain two vertices with the same level but different v-state.*

Proof. In any path, the level of the vertices is non-increasing (by Fact 1.1). Therefore, all the vertices that have the same level must appear consecutively in the path. The lemma follows by Fact 1.5. \square

Lemma 5. *Suppose there exists a reconfiguration sequence ρ of length $2m(n+2)$. Then ρ visits at least one vertex at every level, and all the vertices that it visits at a given level have the same v-state.*

Proof. Since the level of a switched vertex can never increase by more than one, and the level of all vertices in p_b is 0 and in p_e is $i+1$, ρ must visit a vertex at every level.

Let p be the first shortest path in ρ that contains a vertex of level i , and let s be the v-state of that vertex. Let p'' be the first path in ρ after p that does not contain a vertex whose level is i and v-state is s , and let p'

be the path right before that. By Lemma 4, all the paths between p and p' only contain level i vertices whose v-state is s .

Suppose for the sake of contradiction that p'' contains a vertex with level i and a v-state that is not s , and consider the move (a, b, c, d) that created p'' . By Lemma 4 and the fact that p' and p'' differ in only one vertex, we know that b and c are the only level i vertices in p' and p'' , respectively. Furthermore, Fact 1.5 implies all the edges of (a, b, c, d) must be inter-level. We can apply Fact 1.3 to the edges (b, d) , (c, d) (if b is odd) or to (a, b) , (a, c) (if b is even) to show that $cs(b) = cs(c)$. However, since $l(c) = l(b)$, Lemma 3 implies $cs(c) = cs(b) + 1$. This is a contradiction, and, so, p'' does not contain a level i vertex.

For any path, the set of the levels of its vertices must form a contiguous interval, since an edge can never cross more than one level. Because p'' has a level $i + 1$ vertex (by Lemma 3) and does not have a level i vertex, it therefore only contains vertices with level greater than i . Lemma 3 further implies that the same holds for all the paths after p'' in ρ . \square

Suppose there exists a reconfiguration sequence ρ of length $2m(n + 2)$. Lemma 5 allows us to build an assignment θ by assigning θ_i the v-state of the vertices of level i in ρ .

Lemma 6. *The assignment θ is satisfying for ϕ .*

Proof. Consider an arbitrary clause C_j , and the vertices at depth $2j - 1$. Each $p \in \rho$ contains exactly one vertex at this depth. In p_b , the c-state of this vertex is 0, while in p_e it is 1, so there exists some first move (a, b, c, d) with $d(b) = d(c) = 2j - 1$ and $cs(b) = 0$ and $cs(c) = 1$. Since $cs(c) > cs(b)$, Lemma 3 implies that $l(b) = l(c)$. By Fact 1.4, either (b, d) or (c, d) is a formula edge, otherwise we would have $0 = cs(b) = cs(d) = cs(c) = 1$. Furthermore, (b, d) is not a formula edge because $cs(b) = 0$. Thus, (c, d) is a formula edge, and we know from the construction that $x_{l(c)} = vs(c)$ satisfies C_j . But $\theta_{l(c)} = vs(c)$ by definition, so C_j is satisfied. \square

We now prove that the reduction is complete.

Lemma 7. *If ϕ is satisfiable, then there exists a reconfiguration sequence of length at most $2m(n + 2)$.*

Proof. Let $\theta \in \{0, 1\}^n$ be a satisfying truth assignment. Let $sat(i, j) = 1$ if C_j is satisfied by $\theta_1, \dots, \theta_i$, and 0 otherwise. Let $p(i, k) = s$, $v(i, \theta_i, sat(k, 1), 1)$, $v(i, \theta_i, sat(k, 1), 2)$, \dots , $v(i, \theta_i, sat(k, j), 2j - 1)$, $v(i, \theta_i, sat(k, j), 2j)$, \dots , t . We claim that $\rho = p_b, p(1, 0), p(1, 1), \dots, p(i, i - 1), p(i, i), \dots, p_e$ is a reconfiguration sequence of length $2m(n + 2)$, where the intermediate moves and their length are explained below.

The vertices of $p(i, i)$ can be switched in order of increasing depth using inter-level edges to get $p(i + 1, i)$ in $2m$ steps. The paths $p(i, i - 1)$

and $p(i, i)$ are different only when C_j is satisfied by θ_i , in which case there is a formula edge from $v(i, \theta_i, 1, 2j - 1)$ to $v(i, \theta_i, 0, 2j)$. Using this edge, the vertices of $p(i, i - 1)$ can be switched in order of increasing depth to get $p(i, i)$. The number of moves required is $2k$, where k is the number of clauses satisfied by θ_i but not satisfied by $\theta_1, \dots, \theta_{i-1}$. When summed over ρ , these add up to at most $2m$, since each clause can become satisfied for the first time only once. Finally, we can switch between p_b and $p(1, 0)$ and between $p(n, n)$ and p_e using in $2m$ steps each. \square

Combining Lemma 6 and Lemma 7 together with the fact that the reduction can be clearly done in polynomial time, we have the following theorem.

Theorem 2. *The MIN-SPR problem is NP-hard, even if k is polynomial in $|V(G)|$.*

3 Independent set reconfiguration: the models

We now turn our attention to reconfiguring independent sets. Recall that we view an independent set as a set of tokens placed on the vertices such that no two tokens are adjacent. Consider the reconfiguration rule in which a move from one valid configuration to another is made by a *token jump*: moving a token from one vertex to an unoccupied vertex (not necessarily a neighbor of it), such that the resulting set is independent. The token sliding, token jumping, and token addition and removal reconfiguration rules give rise to the following three reconfigurability problems.

Token sliding (TS) / token jumping (TJ): Given a graph G and two independent sets A, B in G , determine if A can be reconfigured into B via a sequence of independent sets, each of which results from the previous one by a single token slide (for TS) or jump (for TJ).

Token addition and removal (TAR): Given a graph G , an integer k and two independent sets A, B in G , both of size $\geq k$, is there a way to transform A into B via independent sets, each of which results from the previous one by adding or removing one vertex of G , without ever going through an independent set of size less than $k - 1$?

We now establish the equivalence between the TJ and TAR problems. We need some definitions. We say that A and B are TS- (TAR-, TJ-) reconfigurable if they belong to the same connected component of the TS- (TAR-, TJ-) reconfiguration graph. For the three independent set reconfiguration problems, we refer to the corresponding reconfiguration graphs as the TS-graph, TAR-graph, and TJ-graph (of the graph G), respectively. Corresponding reconfiguration sequences will be referred to as TS- (TAR-, TJ-) paths. Also, we will use the terms token set and independent set interchangeably.

Theorem 3. *Two independent sets A and B of size s in a graph G are TJ-reconfigurable if and only if they are TAR-reconfigurable with parameter $k = s$. Moreover, $\text{dist}_{\text{TAR}}(A, B) = 2\text{dist}_{\text{TJ}}(A, B)$, and there exists an algorithm that, given a reconfiguration sequence between two independent sets in one of these two models outputs a reconfiguration sequence connecting the two sets in the other model in time polynomial in the length of the sequence. The algorithm maps every shortest TAR-sequence to a shortest TJ-sequence, and vice versa.*

Proof. Any path P in the TJ-graph between two independent sets of size s naturally corresponds to a path P' in the TAR-graph with parameter $k = s$, between the same two sets: Jumping a token from a to b is equivalent to first removing the token from a , thereby creating an independent set of size $s - 1$, and then adding a token to b , resulting in an independent set of size s . Applying this transformation to every jump produces, in linear time, a TAR-reconfiguration path P' such that $|P'| = 2|P|$.

Conversely, suppose that $P = (A_0, A_1, \dots, A_r)$ is a path in the TAR-graph with parameter $k = s$ connecting two independent sets $A = A_0$ and $B = A_r$ of size s , where $A \neq B$. We will show how to transform P into a path P' that connects A to B in the TJ-graph. Notice that this is equivalent to finding an (A, B) -path in the TAR-graph such that all the token sets are of sizes k or $k - 1$.

Let us call an (A, B) -reconfiguration path $P = (A_0, A_1, \dots, A_r)$ in the TAR-graph *compressed* if it holds that $A_i \neq A_{i+2}$ for all $i = 0, 1, \dots, r - 2$. Clearly, every reconfiguration path can be transformed in linear time into a compressed one since if $A_i = A_{i+2}$, then we can remove A_{i+1} and A_{i+2} from the sequence to obtain a shorter TAR-reconfiguration path from A to B . We will refer to this transformation as *compression*.

The whole transformation procedure that will produce a TJ-reconfiguration path will consist of a sequence of compressions and *peak foldings*, which we define now. Let $P = (A_0, A_1, \dots, A_r)$ be a compressed (A, B) -path in the TAR-graph. For each $i \in \{1, \dots, r\}$, let v_i denote the token that is either added or removed in the i -th step, that is, $A_i \triangle A_{i-1} = \{v_i\}$ (where \triangle denotes the symmetric difference). Let us call an index $p \in \{1, \dots, r - 1\}$ *peak index* if

$$|A_p| = |A_{p-1}| + 1 = |A_{p+1}| + 1 \geq k + 1.$$

It follows directly from the definition that for every peak index p , we have $A_p \setminus A_{p-1} = \{v_p\}$ and $A_p \setminus A_{p+1} = \{v_{p+1}\}$. In particular, $v_p, v_{p+1} \in A_p$, and we can obtain another (A, B) -path in the TAR-graph by replacing A_p with the set $A'_p := A_p \setminus \{v_p, v_{p+1}\}$. More formally, *peak folding* is the operation of producing from a compressed path $(A_0, A_1, \dots, A_{p-1}, A_p, A_{p+1}, \dots, A_r)$ the path $(A_0, A_1, \dots, A_{p-1}, A'_p, A_{p+1}, \dots, A_r)$, where p is a peak index and $A'_p = A_p \setminus \{v_p, v_{p+1}\}$. Since P is compressed, the sets A_{p-1} and A_{p+1} are

distinct, which implies that $v_p \neq v_{p+1}$, and in turn $v_{p+1} \in A_{p-1}$. This guarantees that $A'_p = A_{p-1} \setminus \{v_{p+1}\}$ so we can indeed move from A_{p-1} to A'_p with a token removal. Similarly we can see that $A_{p+1} = A'_p \cup \{v_p\}$. Notice that since A'_p is a subset of A_p , it is independent. Figure 3 shows the effect of applying a peak folding on the cardinalities of the token sets.

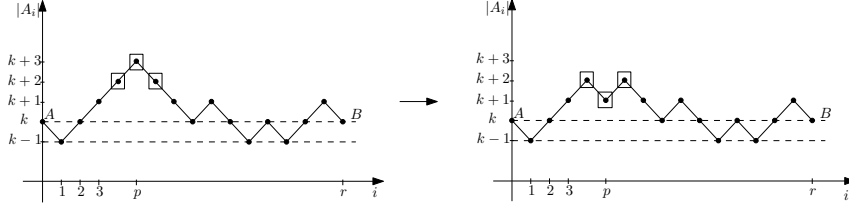


Fig. 3. Simplifying the reconfiguration path by folding a peak at p .

Let $P = (A_0, A_1, \dots, A_r)$ be a given (A, B) -path in the TAR-graph. Starting with P , we iteratively apply compressions and peak foldings (every peak folding preceded by a compression step), as long as possible, that is, until we obtain a compressed path without peak indices. Notice that every step reduces the total size $\sum_i |A_i|$ of the token sets by at least two. All the token set sizes are bounded from below by $k-1$, hence the process eventually stops and produces an (A, B) -path $(A_0^*, A_1^*, \dots, A_m^*)$ in the TAR-graph. Since this path contains no peak indices, it only contains token sets of size k or $k-1$. More specifically, $|A_i^*| = k-1$ for odd i and $|A_i^*| = k$ for even i . In particular, m is even. The path $(A_0^*, A_2^*, A_4^*, \dots, A_{m/2}^*)$ is therefore a valid (A, B) -path in the TJ-graph. Clearly, the algorithm runs in polynomial time.

Finally, it follows from the above proof that shortest paths get mapped to shortest paths. By the first part of the proof, any (A, B) -path of length r in the TJ-graph gets mapped to an (A, B) -path of length $2r$ in the TAR-graph. Notice that every shortest (A, B) -path in the TAR-graph is compressed, thus, by the second part of the proof, every shortest (A, B) -path in the TAR-graph of length r is transformed to an (A, B) -path of length $r/2$ in the TJ-graph. This also shows that $\text{dist}_{\text{TAR}}(A, B) = 2\text{dist}_{\text{TJ}}(A, B)$. \square

Theorem 3 immediately implies that results holding for the TAR model can be transferred to the TJ model. New results can also be derived via this relationship:

Corollary 1. *There exists a polynomial-time algorithm for the TJ problem in line graphs.*

Proof. By Theorem 3, the TJ problem in line graphs is polynomially reducible to the TAR problem in line graphs. Due to the correspondence between matchings in a graph and independent sets in its line graph, the problem is polynomially equivalent to the MATCHING RECONFIGURATION problem. For a polynomial-time algorithm for this problem, see Ito et al. [12]. \square

4 Hardness of independent set reconfiguration

TS, TAR, and TJ reconfiguration problems are all PSPACE-complete in general graphs. For the TS problem, this was announced in [11] (see also [2]) without an explicit proof. For the TAR problem, this was shown by Ito et al. [12]. In fact, their proof uses only token slides (which are done by token additions and removals), also implying that TS is PSPACE-complete. Theorem 3 immediately implies PSPACE-completeness for the TJ model. The hardness of these reconfigurability problems of course implies the hardness of the more difficult problems of finding the length of the shortest reconfiguration sequence. However, in this section we show that these related variants remain NP-hard even when the graph is restricted to be perfect.

We use a reduction from the reconfiguration of shortest paths. Given a graph G and two vertices s and t of G at a distance k apart, let G_1 denote the graph obtained from G by deleting from it all the vertices and edges not appearing on any shortest (s, t) -path. For $i \in \{0, 1, \dots, k\}$, let D_i be the set of vertices in G_1 at distance i from s and $k-i$ from t . The graph G' is the graph obtained from G_1 by turning every set D_i into a clique, and complementing the edges of G_1 between every pair of consecutive layers D_i and D_{i+1} . Formally, $V(G') = V(G_1)$ and $E(G') = \{uv : u \neq v, \exists i \text{ such that } u, v \in D_i\} \cup \bigcup_{i=0}^{k-1} \{uv : u \in D_i, v \in D_{i+1}, uv \notin E(G_1)\}$. The idea of the construction is that there is a bijective correspondence between shortest (s, t) -paths in G and independent sets of size $k+1$ in G' . This gives the following theorem (the proof is straightforward)

Theorem 4. *For every graph G , there is a polynomially computable length-preserving bijection (length-doubling for TAR) between shortest reconfiguration sequences in the shortest path reconfiguration graph of G and those in the TS- (TJ-, TAR-) reconfiguration graph for G' .*

The following corollary is a direct consequence of Theorem 2 and the fact that graph G' contains no odd holes or their complements and hence is perfect [6]. Recall that a *hole* in a graph is a chordless cycle with at least four vertices, and a hole is even (odd) if it has an even (odd) number of vertices.

Corollary 2. *Let G be a perfect graph, A, B two independent sets in G , and k an integer. It is NP-hard to determine if there exists a reconfigu-*

ration sequence of length at most k between A and B in the TS, TJ, and TAR models, even if k is polynomial in $|V(G)|$.

5 Positive results for independent set reconfiguration

In this section we identify two restrictions on the input graphs which make the reconfigurability of independent sets easy to solve.

5.1 Even-hole-free graphs in the TJ model

We will show that two token sets of the same size in any even-hole-free graph are TJ-reconfigurable. Given a graph G and two independent sets A and B in G of the same size, the *Piran graph* $\Pi(A, B)$ of A and B is the subgraph of G induced by the vertex set $(A \setminus B) \cup (B \setminus A)$. The following simple lemma gives a sufficient condition under which it is always possible to jump a token from A to B .

Lemma 8. *Let A and B be two independent sets of the same size in a graph G . If the Piran graph $\Pi(A, B)$ is even-hole-free then there exists a token in $B \setminus A$ with at most one neighbor in $A \setminus B$.*

Proof. The Piran graph is bipartite, and as such, it does not contain odd cycles. If in addition, $\Pi(A, B)$ is even-hole-free, then it must be a forest. Since $|A \setminus B| = |B \setminus A|$, the number of edges in $\Pi(A, B)$ is in fact at most $|A \setminus B| + |B \setminus A| - 1 = 2|B \setminus A| - 1$. Therefore there exists a vertex in $B \setminus A$ with at most one neighbor in $A \setminus B$. \square

A consequence of Lemma 8 is the following result.

Theorem 5. *Let A and B be two independent sets of the same size in a graph G . If the Piran graph $\Pi(A, B)$ is even-hole-free, then A and B are TJ-reconfigurable. Moreover, there exists an algorithm running in time $O(|A|)$ that (if the Piran graph is even-hole-free) finds a shortest TJ-path between the two sets.*

Proof. By Lemma 8, there exists a token in $B \setminus A$ with at most one neighbor in $A \setminus B$. Therefore, the following sequence of token jumps will transform the current independent set A to the target independent set B :

1. Find a vertex v from $B \setminus A$ with at most one neighbor in the set $A \setminus B$.
2. If v has a neighbor in $A \setminus B$, say w , jump w to v .
3. Otherwise, jump an arbitrary token w from $A \setminus B$ to v .
4. Replace A and B with $A \setminus \{w\}$ and $B \setminus \{v\}$, respectively. If $|A| \geq 1$, go to 1.

The whole procedure can be performed in time $O(|A|)$, as follows. We first compute the degrees of the vertices in the $(B \setminus A)$ -part of the initial Piran graph. We keep vertices of degree at most one in a queue. Every token jump consists of taking a vertex v from the queue, finding a vertex $w \in A \setminus B$ as specified above, reducing the Piran graph by exactly two vertices (v and w), and reducing the degrees of the remaining neighbors of w other than v by one. We add each vertex whose degree drops to one to the queue and repeat the procedure. Since every edge of the initial Piran graph is considered at most once, the running time of the algorithm is $O(|E(\Pi(A, B))|) = O(|A|)$. Finally, note that since the reconfiguration sequence output by the algorithm consists of $|A \setminus B|$ moves, it is a shortest one. \square

The class of even-hole-free graphs includes the well known class of chordal graphs (hence also trees and interval graphs). The structure of even-hole-free graphs is understood and membership in this class can be decided in polynomial time [7]. Notice that if the input graph is even-hole-free, so is the Piran graph. Due to Theorem 5 we can easily solve the TJ reconfiguration problem for the class of even-hole-free graphs. Interestingly, determining the complexity of computing the maximum size of an independent set in an even-hole-free graph is, to the best of our knowledge, an open problem.

The example of the claw $G = K_{1,3}$ with leaves $\{v_1, v_2, v_3\}$, and the independent sets $A = \{v_1, v_2\}$, $B = \{v_1, v_3\}$, shows that the analogue of Theorem 5 does not hold for the TS model for the whole class of even-hole-free graphs. We leave it as an open question to determine whether the analogue holds for the class of (claw, even-hole)-free graphs.

5.2 P_4 -free graphs in the TS model

In this subsection we give a polynomial time algorithm to solve the TS problem in P_4 -free graphs. P_4 -free graphs (also known as cographs) are graphs without an induced subgraph isomorphic to a 4-vertex path. A polynomial-time algorithm for token sliding in P_4 -free graphs can be developed based on the following well-known characterization of P_4 -free graphs [8]: a graph G is P_4 -free if and only if for every induced subgraph F of G with at least two vertices, either F or the complement to F is disconnected. A *co-component* of a graph $G = (V, E)$ is the subgraph of G induced by the vertex set of a connected component of the complementary graph $\overline{G} = (V, \{uv \mid u, v \in V, u \neq v, uv \notin E\})$.

Theorem 6. *The TS problem is solvable in time $O(|V| + |E|)$ if the input graph $G = (V, E)$ is P_4 -free. Moreover, a shortest reconfiguration sequence, if it exists, can be found in time $O(|V| + |E|)$.*

Algorithm 1 TS-reconfiguration of independent sets in P_4 -free graphs

Input: A P_4 -free graph $G = (V, E)$ and two independent sets A, B .

Output: A shortest (A, B) -path in the TS-graph, if one exists, NO otherwise.

```

1: if  $|V(G)| = 1$  then return the trivial TS-path if  $|A| = |B|$ , and NO otherwise.
2: if  $G$  is disconnected, with connected components  $C_1, \dots, C_m$  then
3:   if there is an  $i \in \{1, \dots, m\}$  such that  $|A \cap C_i| \neq |B \cap C_i|$  then return NO.
4:   else solve the problem recursively for the connected components  $C_1, \dots, C_m$ 
        with respective token sets  $(A \cap C_1, B \cap C_1), \dots, (A \cap C_m, B \cap C_m)$ .
5:   if one of the outputs is NO then return NO.
6:   else merge the corresponding  $(A \cap C_i, B \cap C_i)$ -paths into an  $(A, B)$ 
        TS-path  $P$ , return  $P$ .
7: else
8:   if  $|A| = |B| = 1$  then return an  $(A, B)$  TS-path corresponding to a shortest
         $(A, B)$ -path in  $G$ .
9:   else
10:    if  $A$  and  $B$  are in the same co-component of  $G$  then solve the problem
        for  $A$  and  $B$  recursively on that co-component and return the output.
11:    else return NO.

```

Proof. We claim that Algorithm 1 below solves the TS problem on P_4 -free graphs.

The correctness of the algorithm is straightforward, using the above-mentioned characterization of P_4 -free graphs [8]. Using the result of Corneil et al. [9] showing that the decomposition of a P_4 -free graph $G = (V, E)$ into one-vertex graphs by means of taking components or co-components can be computed in time $O(|V| + |E|)$ [9], it is also easy to see that the algorithm can be implemented so that it runs in linear time. \square

Theorem 6 can be used to prove that the TS problem is solvable in polynomial time if the input graph is (claw, paw)-free (recall that the claw is $K_{1,3}$ and the paw is the graph obtained from the claw by adding one edge). This is due to the observation that the only connected (claw, paw)-free graph containing an induced P_4 are (long enough) paths and cycles.

6 Concluding remarks

In this paper, we studied the reconfiguration variant of the shortest path problem. We believe that the major open problem is to determine the complexity of deciding whether two shortest paths are reconfigurable. If the problem is NP-hard, then it will be the first example of a “natural” problem in P whose reconfigurability version is NP-hard. If the problem is polynomially solvable, then it will be the first example of an efficiently solvable reconfigurability problem with reconfiguration graphs of large diameter.

Our results are somewhat orthogonal to previous research on reconfiguration since we consider the length of a shortest path between two instances in the reconfiguration graph. If we assume that the bound k on the reconfiguration sequence length is given in unary, MIN-SPR is in NP and Theorem 2 says it is NP-complete. It would be interesting to analyze whether similar results hold for other problems that have been studied in the context of reconfiguration.

Acknowledgements. We are grateful to Takehiro Ito and Daniel Pellicer for interesting and fruitful discussions. We also thank Paul Bonsma for pointing us to problems in P whose reconfiguration versions are PSPACE-complete.

Addendum

Recently, Paul Bonsma proved that shortest path reconfiguration is PSPACE-complete [1].

References

1. Paul S. Bonsma. Shortest path reconfiguration is PSPACE-hard, 2010. arXiv:1009.3217v1 [cs.CC].
2. Paul S. Bonsma and Luis Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theor. Comput. Sci.*, 410(50):5215–5226, 2009.
3. Paul S. Bonsma, Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Finding paths between graph colourings: Computational complexity and possible distances. *Electronic Notes in Discrete Mathematics*, 29:463–469, 2007.
4. Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Connectedness of the graph of vertex-colourings. *Discrete Mathematics*, 308(5-6):913–919, 2008.
5. Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Mixing 3-colourings in bipartite graphs. *European Journal of Combinatorics*, 30:1593–1606, 2009.
6. Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Ann. of Math.*, 164:51–229, 2006.
7. Michele Conforti, Gérard Cornuéjols, Ajtai Kapoor, and Kristina Vučković. Even-hole-free graphs part II: Recognition algorithm. *J. Graph Theory*, 40:238–266, 2002.
8. Derek G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, 1981.
9. Derek G. Corneil, Yehoshua Perl, and Lorna K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14(4):926–934, 1985.
10. Parikshit Gopalan, Phokion G. Kolaitis, Elitza N. Maneva, and Christos H. Papadimitriou. The connectivity of Boolean satisfiability: Computational and structural dichotomies. *SIAM J. Comput.*, 38(6):2330–2355, 2009.
11. Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theor. Comput. Sci.*, 343(1-2):72–96, 2005.
12. Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. In *ISAAC*, volume 5369 of *Lecture Notes in Computer Science*, pages 28–39. Springer, 2008.

13. Takehiro Ito, Marcin Kamiński, and Erik D. Demaine. Reconfiguration of list edge-colorings in a graph. In *WADS*, volume 5664 of *Lecture Notes in Computer Science*, pages 375–386. Springer, 2009.
14. Marcin Kamiński, Paul Medvedev, and Martin Milanič. Shortest paths between shortest paths and independent sets. In *IWOCA*, *Lecture Notes in Computer Science*, volume TBD of *Lecture Notes in Computer Science*. Springer, 2010.